APPENDIX B

IBM PERSONAL COMPUTER DATA ACQUISITION AND CONTROL ADAPTER

This appendix contains the specifications of the analog input and binary output devices. For other device specifications,refer to IBM Personal Computer Data Acquisition and Control Adapter Programming Support. Also, this appendix contains some information about the software and hardware used in this thesis.

FORTRAN FUNCTION LIST

This section contains information on the following functions:

AINM	Analog Input Multiple
AINS	Analog Input Simple
BITOUS	Binary Bit Output Simple
BOUS	Binary Output Simple
DELAY	Delay Execution

Analog Input Device

The analog input device has the following characteristics:

Resolution	12 bits
Input Channels	4 differential
Input Ranges	Switch-selectable ranges: 0 to +10 volts (unipolar), -5 to +5 volts (bipolar), and -10 to +10 volts (bipolar).
Input Resistance	100 megohms minimum
Input Capacitance	200 picofarads maximum; measured at the distribution panel connector
Input Leakage Current	±300 nanoamperes maximum
Input Current	±4 milliamperes at maximum input voltage
Digital Coding	Unipolar: binary. Bipolar: offset binary.
Safe Input Voltage	±30 volts maximum (power On or Off)
Power Supply Rejection	±1/2 LSB maximum change full scale calibration
Integral Linearity Error	±1 LSB maximum

Differential Linearity Error	±1/2 LSB maximum
Differential Linearity Stability	±5 ppm/°C maximum; guaranteed monotonic
Gain Error	±0.1% maximum between ranges. Any range adjustable to zero.
Gain Stability	±32 ppm/°C of FSR maximum
Common-Mode Input Range	±11 volts maximum
Common-Mode Rejection	72 dB minimum ratio (signal within common-mode range)
Unipolar Offset Error	Adjustable to zero
Unipolar Offset Stability	±24 ppm/°C of FSR maximum
Bipolar Offset Error	Adjustable to zero
Bipolar Offset Stability	±24 ppm/°C of FSR maximum

Settling Time	For channel acquisition: 20 microseconds maximum to ±0.1% of the input value
Conversion Time	35 microseconds maximum
Throughput to Memory	15,000 conversions per second, minimum
'A/D convert enable'	
Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor
'A/D convert out'	
Fanout	10 LS TTL loads or 2 standard TTL loads

Binary Output (BO0 through BO15)

Fanout	28 LS TTL loads or 7 standard TTL loads
Throughput from Memory	25,000 operations per second, minimum
'BO Gate'	
Input Impedance	Two LS TTL loads plus one 10-kilohm pull-up resistor
BO CTS	
Input Impedance	One LS TTL load plus 10-kilohm pull-up resistor
'BO Strobe'	
Fanout	10 LS TTL loads or 2 standard TTL loads

Programming with FORTRAN

The FORTRAN bindings are supplied as an object module, DACF.OBJ and DACPF.OBJ. Include the module in the object modules list that the linker requires to make functions accessible to your FORTRAN program.

Editing, Compiling, and Linking

You can create source code for programs in compiled languages by using EDLIN or any other ASCII editor. Call functions just like any other external subroutine. You must observe the variable-declaration, parameter-passing, and array dimensioning conventions of the language.

After compiling the source code, link the resulting object modules with the proper object modules and libraries to form an executable (.EXE) file. Enter the correct one in response to the linker's prompt:

```
DACF.OBJ
```

for FORTRAN Version 2.00 and

```
DACPF.OBJ
```

for Professional FORTRAN.

Once the DAC.COM is loaded, the .EXE files execute in the normal way.

See the IBM Personal Computer *FORTRAN Compiler Version 2.00* or *Professional FORTRAN* for more information on compiling and linking your programs.

Arguments Are...

Every function requires at least one argument; most require several. The argument list determines:

- Which I/O device the function accesses
- On which adapter it is located
- Channel numbers
- The number of samples to read or write
- The variable or array that receives returning input data or sends output data
- The variable that receives the returning execution status.

Most arguments take the form of either integer variables or 2-byte unsigned integers. The language you are using may place other constraints on values or variables. These are explained in the "Remarks" section for each argument.

Types of Arguments

Arguments appear in an argument list following each function. Their purpose determines their place in the list. Not all arguments appear in the argument list of every function; however, the order of the arguments never changes. This order, divided into the following groups, is as follows:

- Adapter, device, and channel numbers. These tell the function which adapter to call. Further, they identify the specific device within that adapter, and the specific channel of that device, if applicable.

- Execution parameters. These supply additional information on execution and data storage.

- Count and rate. These tell iterative functions how many iterations to perform and how fast to perform them.

- Data variable. This is the variable to which an input function writes data, or from which an output function retrieves data.

  Note: In all languages other than C, iterative I/O functions require the data variable to be the first element of a data array.

- Status variable. This is the variable to which the status of the function returns. It indicates the success or type of failure of the function.

When assigning values to arguments (integer arguments in particular), it is important to use the correct data type. It is also important to stay within the appropriate range. To assign a value greater than

32767, convert the unsigned integer value to the signed integer required by BASIC and FORTRAN. Lattice C programmers avoid this problem by using type *unsigned* or type *short integers* for these arguments.

Values greater than 32767, when assigned to integer variables, generate an overflow condition during execution. When returned to integer variables, they usually come out in two's complement form (as values in the range -32768 to -1.) This may affect the way your program tests and uses them.

One way to avoid this is to specify values in hexadecimal form, especially where the bitmasks AND and XOR are concerned. (For more on AND and XOR, see the function pages)

Reading the Argument Pages

The following pages contain detailed descriptions of each argument. In the examples, arbitrary alphabetic labels represent the arguments. You may change them in the code you write. Or, depending on the language you use, you can name them in more or less the same way. They are intended to clarify the purpose of each argument and to indicate its position in the argument list.

Arguments are position-specific. Be sure that arguments for adapter, device, and channel are consistent with the hardware you're accessing. Also make sure that commas (or other recognized delimiters) separate adjacent arguments.

Adapter Number

Label: adapt

Type: Integer value

Range: 0 to 3

Purpose: The adapter number indicates which of the adapters that function accesses.

Remarks: A single Personal Computer can accommodate up to four adapters. Switches on the card assign each an adapter number of 0 to 3. If a value for this argument lies outside this range (or is not assigned to an adapter currently installed in the system), an Unknown Adapter (128) error returns in the status variable.

Related Arguments:

Device Number

Bit Number

Label: bit

Type: Integer value

Range: 0 to 15

Purpose: This argument specifies a binary input or output bit to be tested, set, or cleared by the function.

Remarks: You must assign an integer of value 15 to 0 to this argument. Bit 15 is the most significant bit, and 0 is the least significant. Other values return an Unknown Bit Value (137) error in the status variable.

Channel Low

Label: chanlo

Type: Integer value

Range: 0 to 255

Purpose: This argument selects the channel that the input or output function accesses. In a scanning input function, it specifies the lowest numbered channel included in the scan.

Remarks: Functions accessing a single channel require only a single channel argument. By convention that argument is *chanlo*.

The wide range for this argument provides maximum room for the expansion bus interface. In practice, the accessed device determines the argument's effective range. The allowable values for the on-board devices are:

Device Name	Device #	Channel Range
Analog Input	9	0 to 3
Analog Output	9	0 to 1
Binary I/O	8	Not Applicable
Counter	10	0

For an expansion device, the range of values for this argument depends on the number of channels the device supports. If the value you use is outside the valid channel range for the device, the actual channel selected is determined by the value of *chanlo modulo* the number of channels supported by the device. If *chanlo* is less than 0 or greater than 255, an Invalid Channel Range (134) error returns in the status variable.

Related Arguments:

Channel High

Count

Label: count

Type: Long integer (or real) value

Range: 0 to 16 000 000

Purpose: This determines the number of times an iterative (multiple) function is performed. It also determines the time value of the DELAY function.

Remarks: The value for this argument must not exceed the amount of storage allocated for the target array of the function. It also must not exceed the amount of data in the source array. This is especially true when the function performs a scanning input. These involve count scans, each of which may generate several values.

In Compiled BASIC and Interpreted BASICA, this argument must be a real variable with an integer value in the specified range. Any fractional component is ignored. In C, this argument must be either a variable of type *long int*, or an expression that evaluates to type *long int*. In FORTRAN, this argument must be either a variable of type INTEGER*4 or an expression that evaluates to type INTEGER*4.

If *count* is 0, the function is called but not performed. If *count* is less than 0 or greater than 16 000 000, an Invalid Count Range (135) error returns in the status variable.

Related Arguments:

Data Variable.

Data Variable

Label: data

Type: Integer variable (integer array)

Range: -32768 to 32767

Purpose: This argument references a variable or array element to which a function will write data.

Remarks: If the function is simple (non-iterative), the variable must be an integer variable. If it is a multiple (iterative), or scanning function, the variable must be the first element of an integer array.

The on-board analog input and output devices have a resolution of 12 bits in the range 0 to 4095. Analog output data outside this range is interpreted *modulo* 4096.

The on-board binary and counter timer devices have a resolution of 16 bits; they return data in the range -32768 to 32767. The most significant bit is 15, and the least significant is 0.

Device Number

Label: device

Type: Integer value

Range: 0 to 255

Purpose: This argument determines which I/O device the function accesses. Every I/O device has a unique device number. Each adapter includes the following on-board devices:

Device #	Device
8	Binary I/O device
9	Analog I/O device
10	Counter device

Remarks: As noted above, values for this argument must fall in the range of 8 to 10. Values from 0 to 7 and from 12 to 255 access devices installed through the expansion bus interface. If a value outside this range appears in this argument, an Unknown Device (131) error returns in the status variable.

The device number chosen must correspond to either an adapter installed in the computer or an expansion device. Attempts to access a device that does not exist, or to access a device with an inappropriate function call, can return erroneous values or a Device Timeout (138) error.

Related Arguments:

Adapter Number

Mode

Label: mode

Type: Integer value

Range: 0 or 128

Purpose: This argument determines if system interrupts are enabled or disabled during the processing of multiple I/O functions.

Remarks: This argument applies only to the AINM, AOUM, BINM, BOUM, and CINM multiple I/O functions. Zero is the only allowed value for other functions.

If *mode* is 0, normal system interrupt processing continues during the processing of the multiple I/O functions. If *mode* is 128, the device driver inhibits system interrupts to increase I/O performance.

For values other than 0 and 128, an Unknown Mode (133) error returns in the status variable.

Related Arguments:

Rate.

Rate

Label: rate

Type: Long integer (or real) value

Range: 0 to 1 000 000

Purpose: This specifies the rate, in samples-per-second, at which the function executes.

Remarks: If this value is 0, an external clock signal on the IRQ line determines the sampling rate. Also, at 0, the function does not execute until the IRQ line goes from high to low.

If you enter a value greater than either the function or the current device supports, then iterations occur at the maximum rate. A Timer Overrun (1) error or Excessive Timer Overrun (142) error returns in the status variable.

In Compiled BASIC and Interpreted BASICA programs, this argument must be a real variable containing an integer in the specified range. Decimal components are ignored. In C programs, this argument must be either a variable of type *long int* or an expression that evaluates to type *long int*. In FORTRAN programs, this argument must be either a variable of type INTEGER*4 or an expression that evaluates to type INTEGER*4.

If *rate* is less than 0 or greater than 1 000 000, an Invalid Rate Range (139) error returns to the status variable.

Related Arguments:

Count, Mode

Status Variable

Label: stat

Type: integer

Range: -32768 to 32767

Purpose: This argument references the integer variable to which the function's status code returns.

Remarks: A non-zero return indicates a general execution failure of the function.

Storage Operation

Label: stor

Type: Integer value

Range: 0

Purpose: This argument is reserved.

Remarks: Zero is the only allowed value. For values other than zero, an Unknown Storage Operation (132) error returns in the status variable.

Analog Input Multiple
AINM

Purpose: AINM samples analog values from the specified adapter, device, and channel.

Format: CALL AINM (adapt, device, chanlo, ctrl, mode, stor, count, rate, data, stat)

adapt	Adapter number accessed
device	Device number accessed
chanlo	Channel accessed
ctrl	Expansion device control
mode	Execution mode
stor	Must be 0
count	Number of times the function executes
rate	Execution rate, in samples-per-second
data	First element of the array that receives returning data
stat	Variable that receives the returning execution status.

Analog Input Simple
AINS

Purpose: AINS selects a single analog value from the adapter, device, and channel and stores it in the data variable.

Format: CALL AINS (adapt, device, chanlo, ctrl, data, stat)

adapt	Adapter number accessed
device	Device number accessed
chanlo	Channel accessed
ctrl	Expansion device control
data	Variable that receives the returning data
stat	Variable that receives the returning execution status.

Binary BIT Output Simple
BITOUS

Purpose: BITOUS sets the state of a bit in the binary output word of the adapter and device. The bit takes the value of the data variable.

Format: CALL BITOUS (adapt, device, bit, data, stat)

adapt	Adapter number accessed
device	Device number accessed
bit	The bit number (15 to 0) for output
data	Variable from which the bit value is retrieved (must be 1 or 0)
stat	Variable that receives the returning execution status.

Remarks: When the function is finished, execution status returns to the status variable.

BITOUS neither reads nor affects handshaking lines on the binary input port. The function acts on only the bit specified. It numbers bits from 15 to 0, beginning with the most significant. The single exception to this rule occurs when BITOUS is the first binary output function executed after system initialization. In that case, BITOUS sets or clears the specified bit and zeroes all other bits.

Binary Output Simple
BOUS

Purpose: BOUS outputs the contents of the data variable as a 16-bit binary word.

Format: CALL BOUS (adapt, device, hndshk, data, stat)

adapt	Adapter number accessed
device	Device number accessed
hndshk	Handshake (must be 0)
data	Variable from which data is retrieved
stat	Variable that receives the returning execution status.

Remarks: BOUS operates through the adapter and device. When the function is finished, execution status returns to the status variable. The most significant bit is 15 and the least significant bit is 0.

A data value of 9 (binary word 0000 0000 0000 1001) sets bits 0 and 3 of the binary output word. This latched value remains in effect until changed by another binary output function.

Delay Execution
DELAY

Purpose: DELAY interrupts program execution.

Format: CALL DELAY (adapt, count, stat)

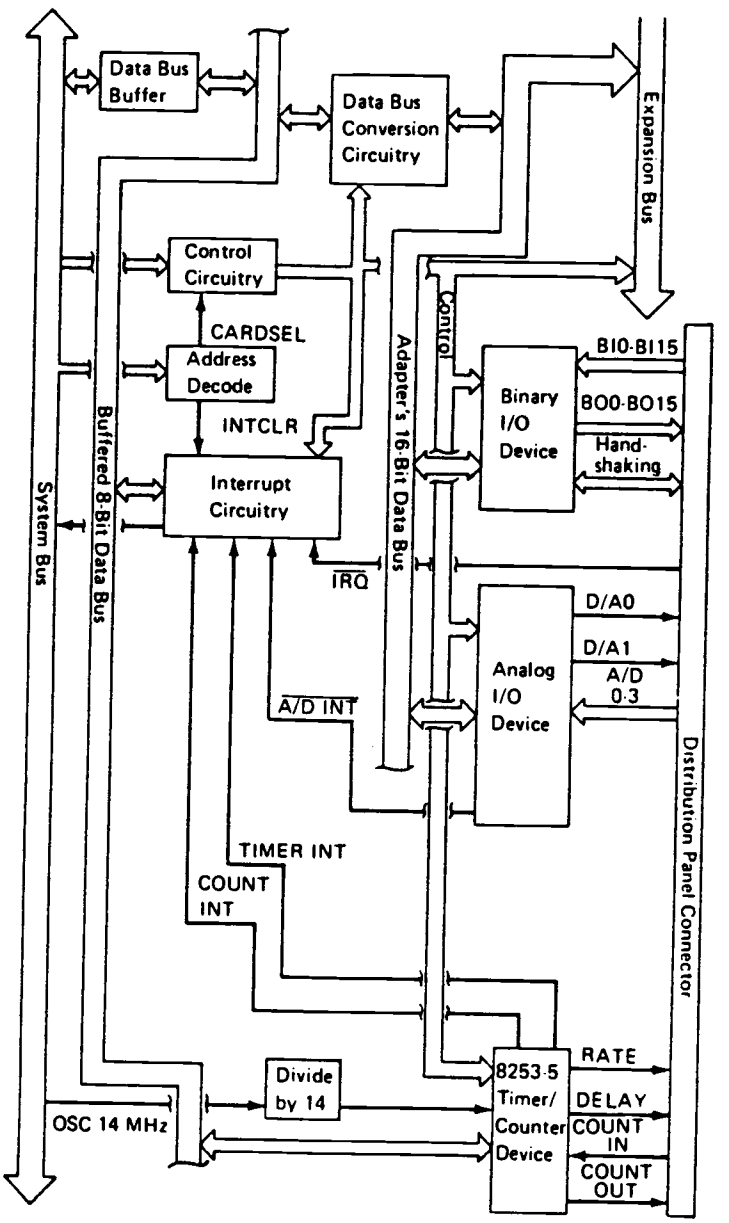
adapt	Adapter number accessed
count	Length of delay (milliseconds)
stat	Variable that receives the returning execution status.

Remarks: DELAY allows you to perform timed sampling at intervals longer than one second allowed by iterative I/O functions. Software overhead must be taken into account. The time required to execute a DELAY function and a subsequent I/O function increases the length of delay by several to several hundred milliseconds.

When the function is finished, execution status returns to the status variable.

Major Components

Following is a block diagram of the Data Acquisition Adapter.



Analog I/O Device

The Data Acquisition Adapter's analog I/O device consists of two subsystems:

- Analog input: An analog-to-digital conversion subsystem.
- Analog output: A digital-to-analog conversion subsystem.

Analog Input Subsystem

On the following page is a block diagram of the analog input subsystem.

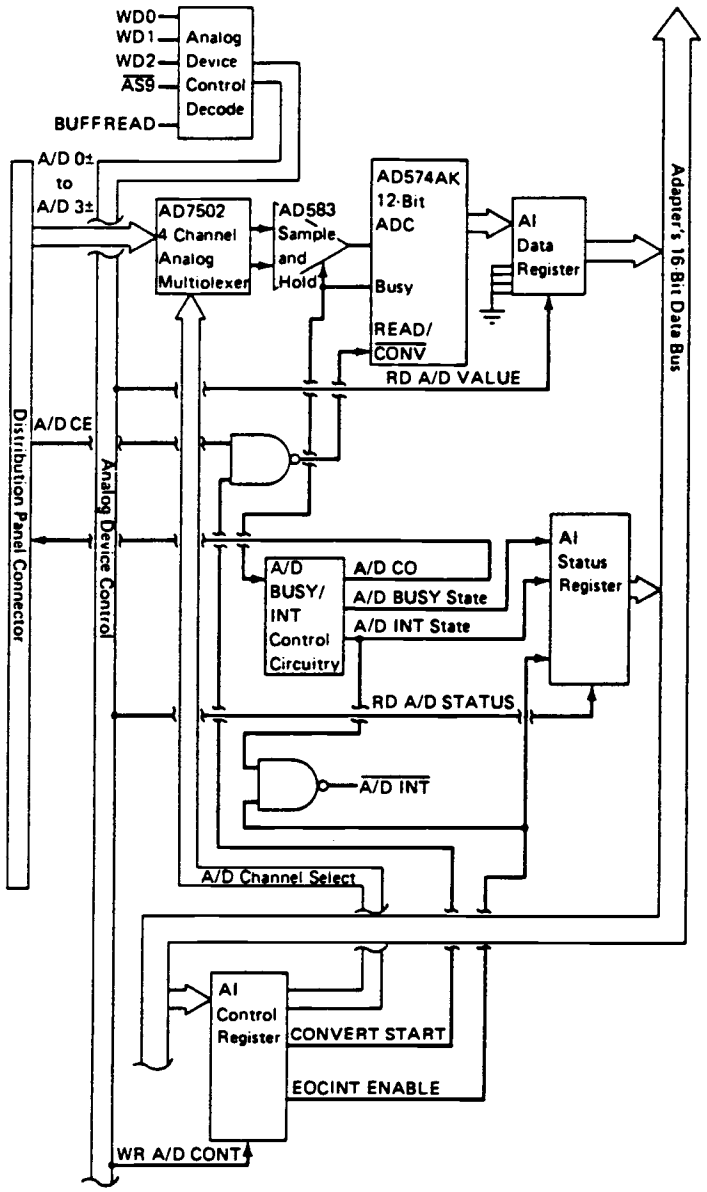
Analog-to-digital conversion is the process of converting analog signals (voltages) over a given range to digital values.

Unlike digital (binary) signals, which have only two voltage states, analog signals have infinite voltage levels over a particular range.

Analog-to-digital converters (ADCs) are categorized by the number of bits of resolution they allow. The greater the number of bits, the greater the number of discrete voltage levels that can be represented.

The Data Acquisition Adapter has an analog input device with the following features:

- Four, multiplexed, differential channels
- An ADC with 12-bit resolution
- Switch-selectable ranges
- Optional data-conversion control with 'A/D convert out' and 'A/D convert enable in' lines.



Analog input subsystem

The Data Acquisition Adapter's analog input device (device number 9) has four channels, which are multiplexed into a single ADC. This device converts analog signals in one of three ranges to digital values in the range of 0 to 4095.

The three switch-selectable ranges are:

- - 5 to +5 volts
- -10 to +10 volts
- 0 to +10 volts

The relationship of the analog input voltage to the returned digital value depends on the range for which the hardware is configured. The selected range setting for analog input is in effect for all analog input channels. For example, in the -5 to +5 volt configuration, an input of +4.997 volts generates a full-scale value of 4095; an input of 0 volts generates a value of 2048; and an input of -5 volts generates a value of 0.

Analog Input Device Control

The use of the $\overline{AS9}$ strobe causes the analog input device to be accessed as device number 9.

The control decode circuitry of the analog device decodes WD0 through WD2, $\overline{AS9}$, and BUFFREAD to generate the following control signals:

WR A/D CONT	Write analog-to-digital control. Allows the AI control register to be written to.
RD A/D STATUS	Read analog-to-digital status. Allows reading of the AI status register.
RD A/D VALUE	Read analog-to-digital value. Allows reading of the AI data register.

Analog Input Device Registers

AI Control Register	The AI control register contains the analog-to-digital channel selection, analog-to-digital interrupt-enable information, and convert start bit information. The AI control register is cleared by $\overline{BUFFRES}$ during power-on- reset.
AI Status Register	The AI status register contains information about 'A/D busy,' the 'A/D interrupt status,' and the readback of the 'A/D interrupt enable.'
AI Data Register	The 16-bit AI data register contains the data from the ADC. Because the output of the ADC is a 12-bit digital value, the four highest bits of the register are grounded.

Starting an Analog-to-Digital Conversion

The convert start bit from the AI control register is logically ANDed with the external 'A/D convert enable' signal from the distribution panel connector. The result is inverted to generate an active low signal, which is brought to the READ/$\overline{\text{CONV}}$ pin of the AD574 ADC.

Reading an Analog-to-Digital Value

The READ/$\overline{\text{CONV}}$ pin must be taken high before the analog-to-digital value can be read. This is accomplished by writing a convert start bit equal to 0 to the AI control register.

Channel Selection

The differential analog to digital channel pair is selected by the AD7502 4-channel, analog multiplexer on the basis of the analog-to-digital channel-select bits of the AI control register.

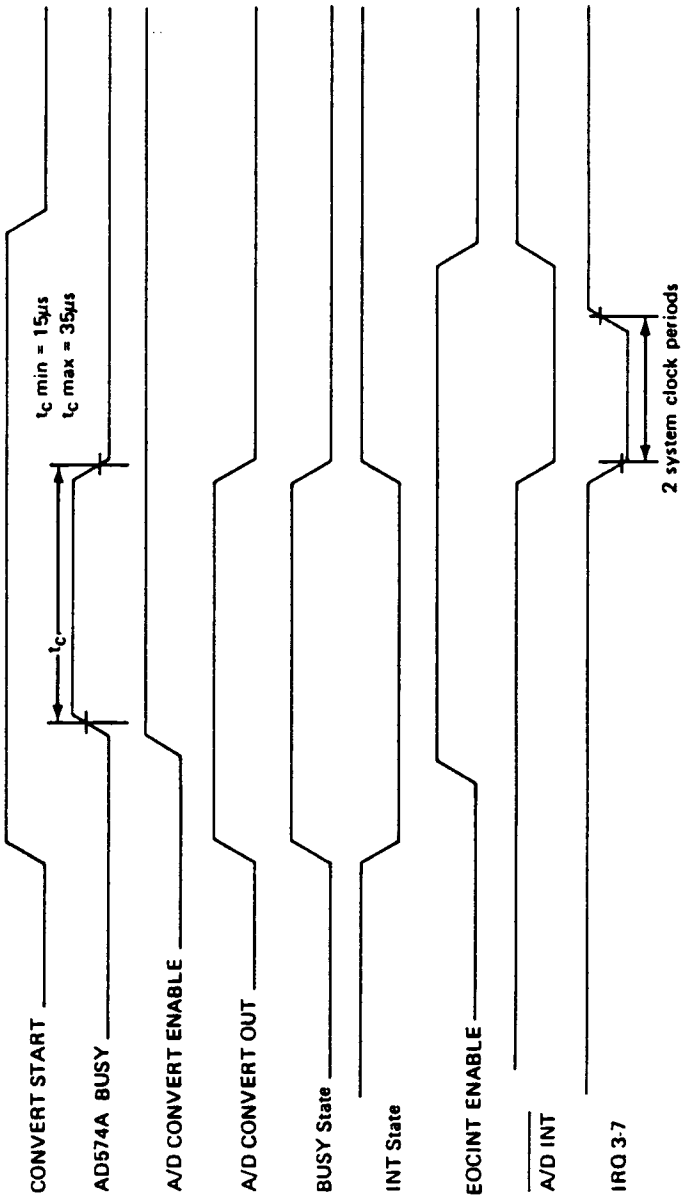
Sample and Hold

During a conversion, the 'busy' signal from the AD574A ADC causes the AD583 Sample and Hold to hold its present value when the 'busy' signal is high, and starts sampling again when it is low.

'A/D Busy' and Interrupt States

At the end of a conversion, the AD574 ADC's 'busy' signal goes low, and the AI status register shows that the analog input device is in the not-busy and interrupting state.

Following is a timing diagram of analog-to-digital conversion.



'A/D Interrupt'

The actual 'A/D interrupt' signal (A/D INT) is a result of the logical ANDing of the INT STATE status bit in the AI status register, and the EOCINT ENABLE bit from the AI control register. The inverted result generates $\overline{\text{A/D INT}}$ (an active low signal), which goes to the interrupt circuitry.

'A/D Convert Out'

The 'A/D convert out' (A/D CO) signal is brought out to the distribution panel connector on the Data Acquisition Adapter.

The 'A/D convert out' signal is set (TTL high) when a conversion has been commanded by programming the convert start bit. The signal remains high until the conversion is complete. If the analog signals received by the on-board analog input device are from an external device that can be made to send data on receipt of a TTL high pulse, you may use a synchronization scheme in which the program's request for an analog-to-digital conversion triggers (using 'A/D convert out') the output of analog data from the external device.

'A/D Convert Enable'

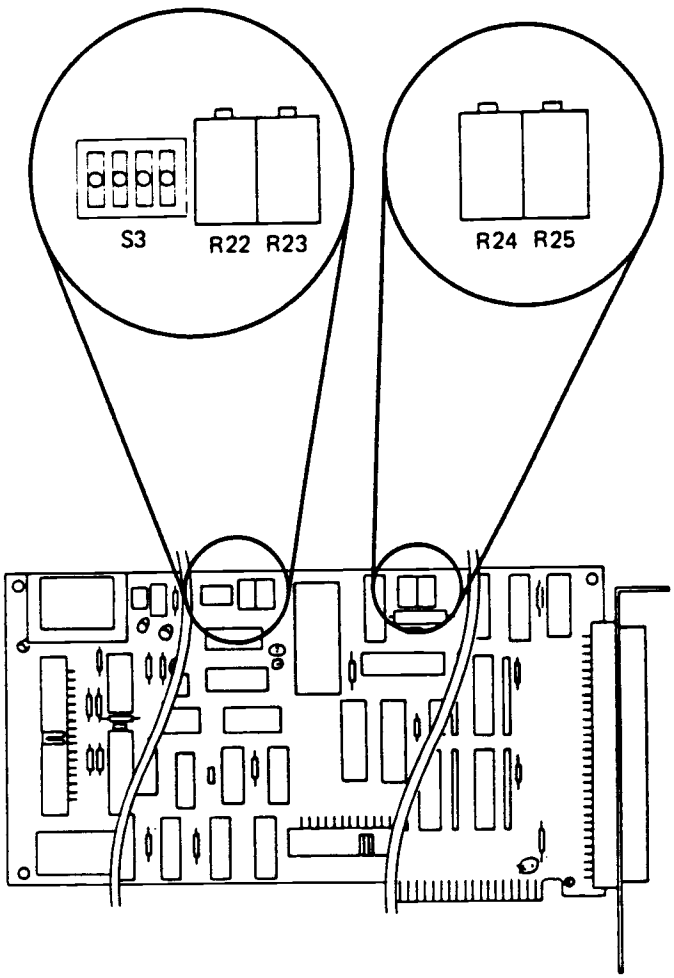
The 'A/D convert enable' (A/D CE) signal is brought out to the distribution panel connector on the Data Acquisition Adapter.

By holding the 'convert enable in' signal low (TTL), an external device can inhibit or delay all analog-to-digital conversions ordered by programming.

To be considered valid and allow an analog-to-digital conversion, the 'convert enable in' signal must remain high until the 'convert out' signal goes low again.

Analog Input Potentiometers

Four potentiometers (R22, R23, R24, and R25) on the Data Acquisition Adapter control bipolar offset, unipolar offset, gain, and common mode rejection for the analog input device. The following diagram shows the location of these potentiometers.



In the following "LSB" represents the weight of the least-significant bit of the 12-bit digital output code of the ADC.

The table shows the 1-LSB values for each analog input range.

Range	1 LSB
0 to +10 volts	2.44 mV
-5 to +5 volts	2.44 mV
-10 to +10 volts	4.88 mV

The ADC is intended to have a 1/2-LSB offset so the exact analog input for a given code will be in the middle of that code (halfway between the transitions to the codes above and below it). The information under "Bipolar Offset" and "Unipolar Offset" explains this 1/2-LSB offset.

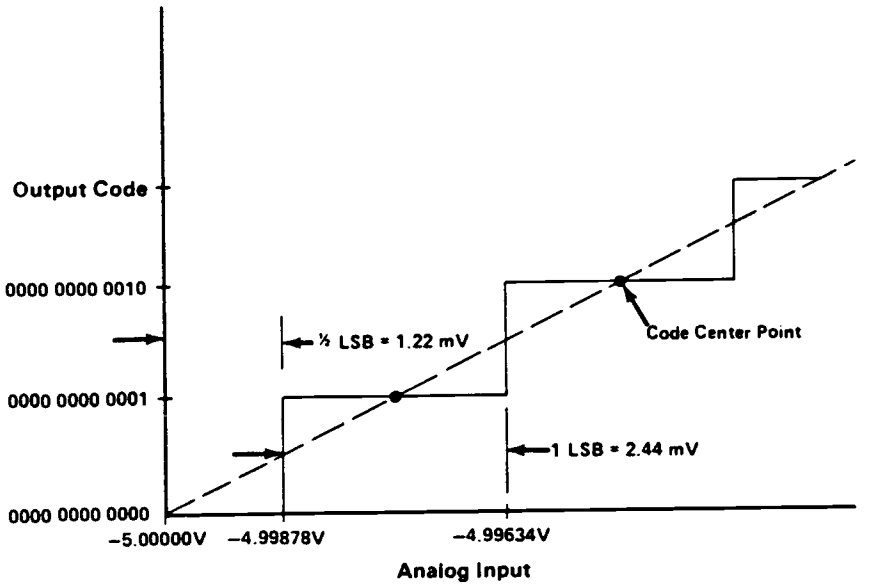
Bipolar Offset:

The value of R22 is set so the transition from the digital output code 0000 0000 0000 to 0000 0000 0001 occurs for an input voltage 1/2 LSB above negative full scale. R22 takes effect when a bipolar range (-5 to +5 volts or -10 to +10 volts) is selected.

The following shows the input voltages for the transition from the output code 0000 0000 0000 to 0000 0000 0001.

Range	Input Voltage for First Code Transition
-5 to +5 volts	-4.99878 volts
-10 to +10 volts	-9.99756 volts

The following shows the first few output-code transitions for the -5 to +5 volt range.



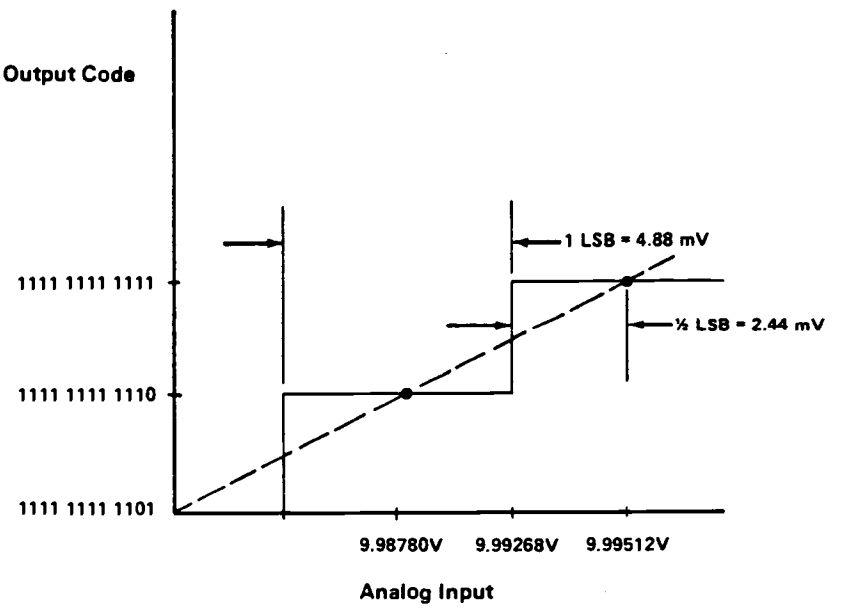
Gain:

The value of R23 is set so the last transition (1111 1111 1110 to 1111 1111 1111) occurs for an input voltage 1-1/2 LSB below full scale.

The following shows the input voltage for the transition from the output code 1111 1111 1110 to 1111 1111 1111.

Range	Input Voltage for Last Code Transition
0 to +10 volts	+9.99634 volts
-5 to +5 volts	+4.99634 volts
-10 to +10 volts	+9.99268 volts

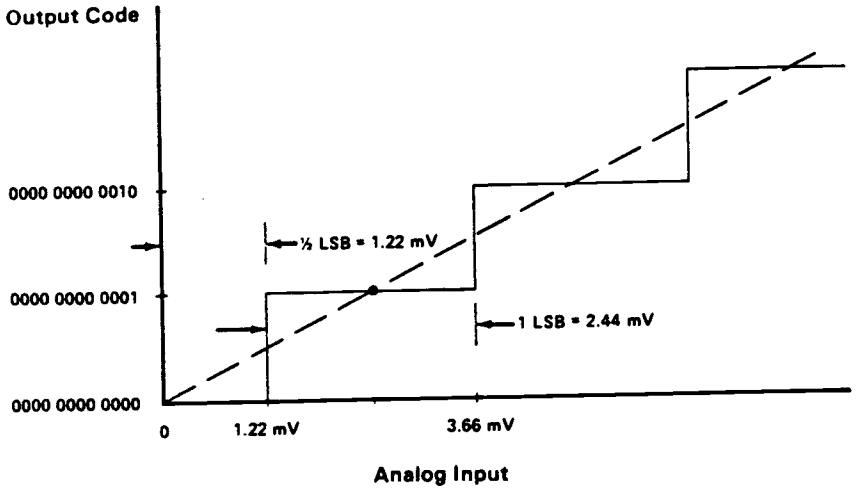
The following shows the last few output-code transitions for the -10 to +10 volt range.



Unipolar Offset:

The value of R24 is set so the first transition (0000 0000 0000 to 0000 0000 0001) occurs for an input voltage of +1/2 LSB. R24 takes effect when the unipolar range (0 to +10 volts) is selected.

The following shows the first few output-code transitions for the 0 to +10 volt range.

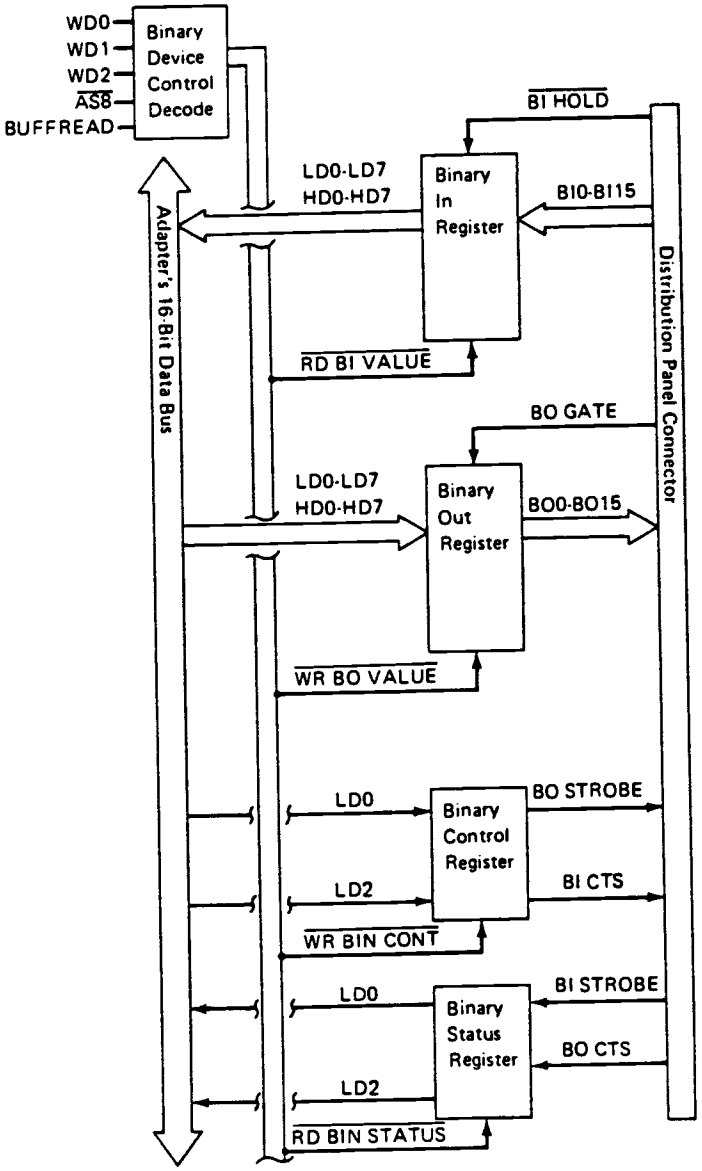


Common Mode Rejection:

R25 allows for the reduction and balancing of the error caused by common mode noise (voltage common to both sides of an analog input channel). The common-mode input range specification for the analog input device is ±11 volts maximum. The value of R25 is set so on the most sensitive range (-5 to +5 volts), the effect of common mode voltage is balanced on each side of zero volts. For example, a common mode voltage of +11 volts produces the same output code as a common mode voltage of -11 volts.

Binary I/O Device

Following is a block diagram of the binary I/O device.



The Data Acquisition Adapter's binary I/O device has the following features:

- A 16-bit binary output port (BO0 through BO15)
- A 16-bit binary input port (BI0 through BI15)
- Input and output handshaking over the 'strobe' and 'clear-to-send' lines
- Direct control using BO GATE ('binary out gate') and $\overline{\text{BI HOLD}}$ ('binary in hold').

Digital signals have only two voltage states: On (high, +3 volts) and Off (low, +0.2 volts). Digital signals in this range are called *TTL signals*, because they are the proper levels to be interpreted by the transistor-to-transistor logic circuitry. These signals have many uses in data acquisition and control applications. Among these are sensing the state of two-state devices and controlling devices that require two-state control signals.

Binary I/O Device Control

The use of the $\overline{\text{AS8}}$ strobe causes the binary I/O device to be accessed as device number 8.

The $\overline{\text{AS8}}$ strobe as an enable, the WD0 through WD2 word bits, and the BUFFREAD signal are used to decode which binary decode operation is to occur.

Following are the four decode operations:

$\overline{\text{WR BIN CONT}}$ — Write binary control: Controls the latching of the binary output strobe (BO STROBE) and the binary input clear-to-send (BI CTS) bits by the binary control register.

$\overline{\text{RD BIN STATUS}}$ — Read binary status: Controls the reading of the binary input strobe (BI STROBE) and the binary output clear to send (BO CTS) bits by the binary status register.

$\overline{\text{WR BO VALUE}}$ — Write binary value: Controls the writing of the binary output word (BO0 through BO15) to the binary output register.

$\overline{\text{RD BI VALUE}}$ — Read binary value: Controls the reading of the binary input word (BI0 through BI15) from the binary input data register.

Binary I/O Device Registers

Following is a description of the binary I/O device registers.

Binary Control Register — Contains the BO STROBE bit and the BI CTS bit. These bits do not physically cause or prevent binary I/O events from occurring. They are programming control bits.

Binary Status Register — Allows the status of BO CTS and BI STROBE bits to be monitored. These bits do not physically cause or prevent binary I/O events from occurring. They are programming status bits.

Binary Input Register — When $\overline{\text{BI HOLD}}$ is brought high (or if no connection is made), the binary input register is not latched and allows the current state of the binary input lines to be monitored by reading the binary input register. Grounding $\overline{\text{BI HOLD}}$ causes the binary input register to latch the current state of all binary input lines. If the grounding of the $\overline{\text{BI HOLD}}$ line is maintained, any later read will obtain the value that was present when the line was initially grounded.

Binary Output Register — Contains the binary output word (BO0 through BO15). Grounding the BO GATE signal places the binary output port in the tri-state condition (all points floating). The binary outputs are gated out when the BO GATE signal is brought high (or if no connection is made).

Binary Output Subsystem

Following is a description of the binary output subsystem.

Binary Output Port (BO0 through BO15)

This subsystem uses high-power, tri-state, bus-driving devices. Changes in the binary output word are carried out on a per-bit basis. Only those bits affected by a change in the output word are actually changed. All others remain the same.

The output port of the binary I/O device supplies 16 high/low signals under program control. As with the input port, these signals can be used individually or considered as a 16-bit data word.

Binary Out Gate

You may place the output port in tri-state by pulling the binary out gate (BO GATE) lines low. These and all other data, handshaking, and control lines are pulled high by internal resistors to +5 volts. No connections to them are necessary unless your application requires handshaking or control.

Binary Output Handshaking

Because all communication lines are internally pulled up to their logical true state, you can use or not use binary output handshaking, depending on the requirements of your communication setup.

Binary output can be synchronized with the data input capabilities of the external device. The external device must be able to send a TTL signal to indicate it is ready for new data. It also must be able to accept parallel binary data when it recieves a signal from the Data Acquisition Adapter's binary I/O device indicating the data is available.

Error Codes

Errors return in decimal to the status variable. Check this variable after an execution to see if an error has occurred. All errors return in the same way. There is, however, a difference between "hard" and "soft" errors, as explained below.

No Error

You see a No Error condition reported in the status variable when everything is working properly.

Error Code	Description
0	No Error: The function has executed normally and control returns to the caller.

Soft Error

A soft error allows the function to execute, but affects the integrity of the data.

Error Code	Description
1	Timer Overrun: The execution rate was faster at times than the device and function could manage. The function executed, but a small percentage of samples was taken at an irregular and slower rate than specified.
142	Excessive Timer Overrun: The execution rate was often faster than the device and function could manage. The function executed, but a large percentage of samples was taken at an irregular and slower rate than specified.

Error Code	Description
135	Invalid Count Range: An incorrect value is specified for the count argument.
136	Unknown Handshake Value: An incorrect value is specified for the handshake argument.
137	Unknown Bit Value: The value for the bit number argument is outside the range of valid bits.
138	Device Timeout: A device that does not exist was specified for the device argument or the external handshaking did not occur.
139	Invalid Rate Range: The value for the rate argument is outside the valid range.

Hard Errors

A hard error indicates a failure to execute. The function ends prematurely and control returns to the caller. No data is collected.

Error Code	Description
-2	No Device Driver: (Compiled languages only). The device driver DAC.COM was not found.
128	Unknown Adapter: The requested adapter is either not in the system (that is, no adapter is addressed to that adapter number), or the adapter is not working.
131	Unknown Device: Either the requested device is not known, or a presence test on the device has failed.
132	Unknown Storage Operation: An incorrect value is specified for the storage argument.
133	Unknown Execution Mode: An incorrect value is specified for the execution mode argument.
134	Invalid Channel Range: Values for the channel low and channel high arguments do not set a valid channel range (that is, *chanhi* has a lower value than *chanlo* or is greater than 255).

APPENDIX C

PROGRAMS

```
C     ************************POWER*****************************
C     *                                                        *
C     * THIS PROGRAM MEASURES OUTPUT VOLTAGE FROM A WIDE RANGE *
C     * LOG-CHANNEL AMPLIFIER  WHICH IS RELATED TO  THE  POWER *
C     * OF  OREGON  STATE  UNIVERSITY  TRIGA  REACTOR  (OSTR), *
C     * 10 VOLTS CORRESPONDS TO 1 MEGA WATT.                   *
C     * THE REACTOR POWER IS READ FROM  THE LINEAR CHANNEL  TO *
C     * GET A MORE ACCURATE EQUATION BETWEEN REACTOR POWER AND *
C     * THE ADC OUTPUT CODE (VOLTAGE).  THE POWER IS  MEASURED *
C     * TEN  TIMES  FOR  EACH  CASE  USING ANALOG INPUT SIMPLE *
C     * FUNCTION  (AINS),AND AN AVERAGE  OF THESE MEASUREMENTS *
C     * IS TAKEN.                                              *
C     ************************************************************
      PROGRAM POWER
      INTEGER*2 ADAPT,DEVICE,CHANLO,CTRL,
     *RAWVAL,STAT,I
      CHARACTER*14 ANS,DATAOUT
      ADAPT=0
      DEVICE=9
      CHANLO=3
      CTRL=0
      STAT=0
      I=0
      WRITE(*,5)
 5    FORMAT(' ENTER OUTPUT DATA FILE NAME '/)
      READ(*,1)DATAOUT
 1    FORMAT(A14)
      OPEN(UNIT=81,FILE=DATAOUT,STATUS='NEW')
 7    WRITE(*,10)
 10   FORMAT(' ENTER R WHEN YOU ARE READY'/)
      READ(*,1)ANS
      IF(ANS.EQ.'R'.OR.ANS.EQ.'r') GOTO 15
      GOTO 7
 15   P1=0.0
      DO 150 J=1,10
      CALL AINS(ADAPT,DEVICE,CHANLO,CTRL,
     *RAWVAL,STAT)
      IF(STAT.NE.0) GOTO 300
      P1=RAWVAL+P1
 150  CONTINUE
      P1=P1/10.0
      WRITE(81,100)P1
      WRITE(*,100)P1
 100  FORMAT(1X,F10.0)
      WRITE(*,20)
 20   FORMAT(' ENTER Y WHEN YOU HAVE  OTHER DATA OR ANY
     *CHERACTOR IF NOT '/)
      READ(*,1)ANS
      IF(ANS.EQ.'Y'.OR.ANS.EQ.'y') GOTO 7
      GOTO 400
 300  WRITE(*,200)STAT
 200  FORMAT(1X,'EXECUTION ERROR',I6/)
 400  CLOSE(81)
      STOP
      END
```

```
C     ************************CURVE*********************************
C     *                                                            *
C     * THIS  PROGRAM  IS USED TO  FIND  AN  EQUATION  BETWEEN *
C     * THE  REACTOR  POWER AND THE MEASURED ADC OUTPUT  CODE. *
C     * SINCE  THE  CODE IS PROPORTIONAL  TO THE LOGARITHM  OF *
C     * REACTOR POWER , THE LOGARITHM  OF THE POWER  IS TAKEN, *
C     * AND LINEAR REGRESSION IS USED TO FIND THE EQUATION.    *
C     * THE EQUATION IS IN THE FORM OF ( POWER = 10**(A+B*Cp)) *
C     * WHERE Cp IS THE CODE ASSOCIATED WITH REACTOR POWER.    *
C     ************************************************************
      DIMENSION X(100),Y(100),Z(100),R(100)
      SUM0=0.0
      SUM1=0.0
      SUM2=0.0
      SUM3=0.0
      SUM4=0.0
      PRINT*,' ENTER THE NUMBER OF DATA '
      READ(*,1)I
 1    FORMAT(I3)
      DO 10 J=1,I
      READ(*,2)X(J),Y(J)
 2    FORMAT(2F10.0)
      Z(J)=ALOG10(Y(J))
 10   CONTINUE
      WRITE(*,3)(X(J),Y(J),J=1,I)
 3    FORMAT(1X,2F15.5)
      DO 20 JJ=1,I
      R(JJ)=Z(JJ)*X(JJ)
 20   CONTINUE
      DO 30 II=1,I
      SUM0=SUM0+R(II)
      SUM1=SUM1+X(II)
      SUM2=SUM2+Z(II)
      SUM3=SUM3+X(II)*X(II)
      SUM4=SUM4+Z(II)*Z(II)
 30   CONTINUE
      B=(SUM0-SUM1*SUM2/I)/(SUM3-SUM1*SUM1/I)
      A=(SUM2/I-B*SUM1/I)
      XR=(SUM0-SUM1*SUM2/I)**2/((SUM3-SUM1*SUM1/I)*(SUM4-SUM2*
     *SUM2/I))
      WRITE(*,40)A,B,XR
 40   FORMAT(1X,'A= ',F15.5,1X,'B= ',F15.5,1X,'R*2 = ',F15.10)
      END
```

```
C   ***************************LOGGER*************************
C   *                                                        *
C   * WRITTEN BY                                             *
C   *            ALLA J. MOHHAMMAD BAKIR                     *
C   *            COLLEGE OF ENGINEERING                      *
C   *            NUCLEAR ENGINEERING DEPTARTMENT             *
C   *            OREGON STATE UNIVERSITY                     *
C   *            MAY 1 , 1988                                *
C   * THIS  PROGRAM  IS  DERIVED FROM A PROGRAM WRITTEN BY:  *
C   * ROBERT J. TUTTLE,ATOMICS INTERNATIONAL, MARCH 1,1967.  *
C   * THIS PROGRAM MEASURES  A  VOLTAGE  THAT IS RELATED TO  *
C   * REACTOR  POWER  USING  DATA  ACQUSITION  AND  CONTROL  *
C   * ADAPTER , AND  FROM THE  REACTOR  POWER  HISTORY  THE  *
C   * REACTIVITY CAN BE CALCULATED USING THE POINT KINETICS  *
C   * EQUATIONS.                                             *
C   * INPUT DATA CONSIST OF THE FRACTIONAL RELATIVE DELAYED  *
C   * NEUTRON YIELDS(A(I),SUM OF A(I)=1.0),DELAYED  NEUTRON  *
C   * PRECURSOR  DECAY  CONSTANTS ( LAMBDA(I) IN RECIPROCAL  *
C   * SECOND), THE PROMPT NEUTRON LIFETIME ( L IN SECONDS),  *
C   * THE  EFFECTIVE  DELAYED  NEUTRON FRACTION (BETA), THE  *
C   * VALUE  OF  THE  SOURCE  CORRECTION TERM (SC IN CENTS/  *
C   * SECOND/COUNT).                                         *
C   * THIS PROGRAM USES A SPECIAL FUNCTION  SUBROUTINE, FOR  *
C   * MORE INFORMATION  ABOUT THESE  FUNCTIONS REFER TO THE  *
C   * IBM  PERSONAL  COMPUTER  DATA ACQUISITION AND CONTROL  *
C   * ADAPTER PROGRAMMING SUPPORT.                           *
C   * REACTOR  POWER  SHOULD BE CONSTANT FOR ABOUT 1 MINTUE  *
C   * BEFORE  DATA  COLLECTION  BEGINS  AND  NO  REACTIVITY  *
C   * CHANGE SHOULD BE MADE DURING THIS PERIOD , THE  POWER  *
C   * DATA IS COLLECTED BY A RATE=RATE1, AND THE COLLECTION  *
C   * TIME  IS  EQUAL  TO   COUNT1/RATE1  WHICH IS EQUAL TO  *
C   * 1 MINUTE, THIS WAITING TIME PERMITS ESTABLISHMENT  OF  *
C   * THE   EQUILIBRIUM   PRECURSOR   POPULATION  WHICH  IS  *
C   * CALCULATED  FROM  THE  AVERAGE POWER IN THE FIRST ONE  *
C   * MINUTE.                                                *
C   * REACTOR POWER FOR WHICH THE REACTIVITY TO BE FOUND IS  *
C   * MEASURED  BY A  RATE=RATE2, AND  COLLECTION  TIME  IS  *
C   * EQUAL TO COUNT2/RATE2.                                 *
C   * THIS  PROGRAM  HAS  THREE  SECTIONS, FIRST IS A POWER  *
C   * MEASURING SECTION, IN WHICH A VOLTAGE RELATED TO  THE  *
C   * REACTOR  POWER  IS MEASURED BY THE DATA ACQUISION AND  *
C   * CONTROL  ADAPTER, A SECOND  SECTION  INVOLVED  WITH A  *
C   * REACTIVITY   CALCULATION ,  AND  FINALLY  DISPLAY  OF  *
C   * REACTIVITY ON A SEVEN SEGMENT DISPLAY.                 *
C   * ********************************************************
      PROGRAM LOGGER
      DIMENSION Y1(3000),Y2(3000),PIN(3000),XB(3000)
      INTEGER*2 RAWDTA(3000),POWDTA(3000)
      INTEGER*2 ADAPT,DEVICE,CHANLO,CTRL,MODE,
     *STOR,STAT
      INTEGER*4 COUNT1,RATE1,COUNT2,RATE2
      INTEGER*4 COUNT,RATE
      REAL A(6),LAMBDA(6),L,T(3000),R(3000)
      DOUBLE PRECISION
     1B(6),BD(6),C(6),D(6),X1(6),K(3000),
     2AT,DN,P0,DP,PBAR,SC,SUM
      CHARACTER*14 DATAOUT,FILENAME,RECIN
      ADAPT=0
      DEVICE=9
      CHANLO=3
```

```
      CTRL=0
      MODE=0
      STOR=0
      STAT=0
      READ(*,10)(A(I),I=1,6)
   10 FORMAT(6F10.0)
      READ(*,10)(LAMBDA(I),I=1,6)
      READ(*,20)L,BETA,SC
   20 FORMAT(3F10.0)
      WRITE(*,1)
    1 FORMAT(' ENTER COUNT1, RATE1 2I4 FORMAT '/)
      READ(*,2)COUNT1,RATE1
    2 FORMAT(2I4)
      WRITE(*,3)
    3 FORMAT(' ENTER COUNT2,RATE2 2I4 FORMAT '/)
      READ(*,2)COUNT2,RATE2
      WRITE(*,4)
    4 FORMAT(' ENTER OUTPUT DATA FILENAME '/)
      READ(*,5)DATAOUT
    5 FORMAT(A14)
      WRITE(*,6)
    6 FORMAT(' ENTER REC. FILENAME '/)
      READ(*,5)FILENAME
      WRITE(*,7)
    7 FORMAT(' ENTER POWER FILENAME AND BE READY '/)
      READ(*,5)RECIN
      OPEN(UNIT=81,FILE=DATAOUT,STATUS='NEW')
      OPEN(UNIT=10,FILE=FILENAME,STATUS='NEW')
      OPEN(UNIT=90,FILE=RECIN,STATUS='NEW')
      COUNT=COUNT1
      RATE=RATE1
C
C     POWER MEASURING SECTION
C
      CALL AINM(ADAPT,DEVICE,CHANLO,CTRL,
     *MODE,STOR,COUNT,RATE,RAWDTA(1),STAT)
      IF(STAT.NE.0) GOTO 300
      COUNT=COUNT2
      RATE=RATE2
      CALL AINM(ADAPT,DEVICE,CHANLO,CTRL,
     *MODE,STOR,COUNT,RATE,POWDTA(1),STAT)
      IF(STAT.NE.0) GOTO 300
      WRITE(81,100)(RAWDTA(I),I=1,COUNT1)
      WRITE(81,100)(POWDTA(I),I=1,COUNT2)
  100 FORMAT(1X,5I10)
      CLOSE(81)
C
C     REACTIVITY CALCULATION SECTION
C
      DT=1.0/RATE2
      NOP=COUNT2
      DO 200 I=1,COUNT1
      IF(RAWDTA(I).GE.0.AND.RAWDTA(I).LE.2488)THEN
      Y1(I)=10.**(-4.06015+0.00245*RAWDTA(I))
      ELSEIF(RAWDTA(I).GT.2488.AND.RAWDTA(I).LT.2680)THEN
      Y1(I)=10.**(-3.81625+0.00233*RAWDTA(I))
      ELSEIF(RAWDTA(I).GE.2680)THEN
      Y1(I)=10.**(-4.26878+0.0025*RAWDTA(I))
      ENDIF
  200 CONTINUE
```

```
      DO 350 I=1,COUNT2
      IF(POWDTA(I).GE.0.AND.POWDTA(I).LE.2488)THEN
      Y2(I)=10.**(-4.06015+0.00245*POWDTA(I))
      ELSEIF(POWDTA(I).GT.2488.AND.POWDTA(I).LT.2680)THEN
      Y2(I)=10.**(-3.81625+0.00233*POWDTA(I))
      ELSEIF(POWDTA(I).GE.2680)THEN
      Y2(I)=10.**(-4.26878+0.0025*POWDTA(I))
      ENDIF
  350 CONTINUE
      AT=L/BETA/DT
      SUM1=0.0
      DO 400 I=1,COUNT1
      SUM1=SUM1+Y1(I)
  400 CONTINUE
      C(1)=SUM1/COUNT1
      SC=SC*DT/100.0
      R(1)=-SC/C(1)
      K(1)=1.0+BETA*(1.0+BETA*R(1))*R(1)
      T(1)=DT
      DO 1100 I=1,6
      X1(I)=DBLE(LAMBDA(I)*DT)
      B(I)=DEXP(-X1(I))
      IF(X1(I).GT.0.1) GOTO 1000
      BD(I)=X1(I)-0.5*X1(I)**2+0.162582*X1(I)**3
      D(I)=0.5*X1(I)-0.166666*X1(I)**2+0.041847*X1(I)**3
      GOTO 1100
 1000 BD(I)=1.0D00-B(I)
      D(I)=(X1(I)-BD(I))/X1(I)
 1100 C(I)=K(1)*C(1)
      N1=NOP-2
      DO 1300 N=2,N1
      K(N)=1.0+BETA*K(N-1)*R(N-1)
      SUM=0.0
      DP=0.5*(Y2(N+1)-Y2(N))
      P0=Y2(N)-0.5*DP
      DO 1200 I=1,6
      C(I)=C(I)*B(I)+K(N)*(Y2(N)*BD(I)+2*DP*D(I))
 1200 SUM=SUM+A(I)*C(I)
      PBAR=(Y2(N+1)+Y2(N))/2.0
      PIN(N)=DT/PBAR
      DN=Y2(N+1)-Y2(N)
      R(N)=1.0+(AT*DN-SUM-SC)/(PBAR*K(N))
      K(N)=1.0+BETA*K(N)*R(N)
      R(N)=1.0D00+(AT*DN-SUM-SC)/(PBAR*K(N))
      WRITE(*,500)R(N)
  500 FORMAT(1X,E16.6)
 1300 T(N)=T(N-1)+DT
      WRITE(10,19)(T(N),R(N),N=1,N1)
 19   FORMAT(1X,F10.5,2X,E16.6)
      WRITE(90,19)(T(N),Y2(N),N=1,N1)
      CLOSE(10)
      CLOSE(90)
C
C     REACTIVITY DISPLAYING SECTION
C
      COUNT=1000
      BIT=0
      HNDSHK=0
      DEVICE=8
      DO 205 I=1,N1
```

```
      XB(I)=ABS(R(I))
      NX=1000*XB(I)
      NX1=NX/1000
      NX2=NX/100-NX1*10
      NX3=NX/10-10*NX2-100*NX1
      NX4=NX-10*NX3-100*NX2-1000*NX1
      RAWVAL=NX1*4096+NX2*256+NX3*16+NX4
      CALL BOUS(ADAPT,DEVICE,HNDSHK,RAWVAL,
     *STAT)
      IF(STAT.NE.0) GOTO 300
      IF(R(I).LT.0.0) THEN
      RAW=1
      GOTO 405
      ELSE
      RAW=0
      ENDIF
  405 CALL BITOUS(ADAPT,DEVICE,BIT,RAW,STAT)
      IF(STAT.NE.0) GOTO 300
      CALL DELAY(ADAPT,COUNT,STAT)
      IF(STAT.NE.0) GOTO 300
  205 CONTINUE
  300 WRITE(*,150)STAT
      WRITE(*,102)
  102 FORMAT(1X,' EXECUTION COMPLETE. ',/)
      GOTO 101
  150 FORMAT(1X,' EXECUTION ERROR ',I6)
  101 STOP
      END
```